METHOD OF PROVIDING DATABASE FUNCTIONS FOR MULTIPLE INTERNET SOURCES

RELATED APPLICATIONS

This application is related to commonly assigned United States Provisional Patent Application serial no. 60/112,769, filed December 18, 1998 entitled "METHOD OF PROVIDING DATABASE FUNCTIONS FOR MULTIPLE INTERNET SOURCES" and commonly assigned United States Provisional Patent Application serial no. 60/147,875, filed August 9, 1999 and entitled "EXTENDING BROWSER FUNCTIONALITY BEYOND HTML PAGE PRESENTATION".

FIELD OF THE INVENTION

The present invention relates to web browsers for the Internet and more particularly to a database utility therefore.

BACKGROUND OF THE INVENTION

The number of users professionally using the Internet (and particularly the "World Wide Web") as a data source, and hence analogously to a database, on a daily basis is becoming increasingly greater. However, most web operations are largely performed manually. Gleaning relevant information from individual web pages is tedious. In order to provide useful applications of information accessible through the Internet often requires consolidation of data from multiple sources. This is true of the input side, for example, in uniform resource locator ("URL") specifications, login names, password and other access codes, profiles, queries, etc., as well as on the output side, for example search results, data

extraction from a web page, composition, editing and further processing. Professional web users are currently lacking tools that are standard on modern databases, and accordingly, a substantial amount of time is spent performing mundane manipulations.

One of the reasons why standard database tools can not readily be used on the web is the fact that there is no standardized way to access information, largely because web pages are designed primarily for human, and not machine readability for example. Further exasperating the situation, data is typically not stable; i.e. even if the core information of a page remains the same, presentation and therefore the coding of a page can change arbitrarily often, thus defeating any hard coded access, search or retrieval method. Accordingly, there is a need for, and an object of the present invention, to overcome these problems and provide a data location and extraction tool capable of automated operation. A further object of the present invention is to automatically provide a computerized method and system capable of automatically navigating to a plurality of destination websites, extracting select pieces of data therefrom, processing the extracted data and displaying the processed data in an organized format.

SUMMARY OF THE INVENTION

A method for automatically extracting data from at least one electronic document being accessible over a computer network, the method including: recording a sequence of actions operable to electronically navigate to a target page of the electronic document, the target page including a plurality of elements each having a structural definition wherein the structural definitions interrelate the plurality of elements; identifying a target pattern for a selected subset of the plurality of elements; automatically accessing the target page

according to the recorded sequence; and, automatically identifying and copying select ones of the plurality of elements and the structural definitions of the select ones of the plurality of elements dependent upon the target pattern.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates an overview of the present invention.

Figure 2 illustrates an overview of recording a navigation script.

Figure 3 illustrates an overview of the storage XML files and various variations in links utilized in the preferred form of the invention.

Figure 4 illustrates an overview of the recording extraction script according to a preferred form of the present invention.

Figure 5 illustrates the use of structural and contents spaces according to the preferred embodiment of the present invention.

Figure 6 illustrates an overview of the extraction playback module according to the preferred embodiment of the present invention.

Figure 7 illustrates an overview of a system integrating the preferred embodiment of the present invention.

Figure 8 illustrates a step of a user surfing to a desired page and recording the necessary navigation steps to get there according to the preferred form of the present invention.

Figure 9 illustrates a user highlighting text to be extracted from a web page according to the preferred form of the present invention.

Figure 10 illustrates text which has been automatically extracted according to the preferred form of the present invention.

Figure 11 illustrates the use of a find text command according to the preferred form of the present invention.

Figure 12 illustrates the use of a web page hierarchy according to the preferred form of the present invention.

Figure 13 illustrates the use of a web page hierarchy to determine an extraction pattern according to the preferred form of the present invention.

Figure 14 illustrates a criteria pattern selection screen according to the present invention.

Figure 15 illustrates how macro steps can be used to incorporate three type of pattern matching and extraction according to the present invention.

Figure 16 illustrates a web band incorporated into commercially available web browsing software according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

According to the present invention and referring now to the numerous figures wherein like references identify like elements of the invention, one can automatically navigate to a plurality of destination websites, extract specified information based upon taught schemas, process the extracted data according to customizable scripts, integrate information from other applications such as Microsoft Word, Excel and Access, view the final output using a browser such as Microsoft Internet Explorer for example and automatically repeat these steps in a scheduled manner or when requested for example.

Referring now to Figure 1, the present invention represents a core layer on top of which automated web applications can be built. In other words, the present invention represents a tool, which can be used by developers, or web users utilizing a proper interface, to access data available within one or more web pages in an automated fashion. A navigation Application Program Interface ("API") 10 allows a client application program running on a user's microprocessor based device to learn and store navigation paths to given web pages, including dialogs and forms that need to be filled out to reach those locations or sites for example. The navigation API 10 includes a recording module 12 and playback module 14. For instance, if a site requires a user to enter a login name and password to reach an orientation page, then asks for a set of preferences to go to specific pages of interest, it is an object of the present invention to enable a client application to record this path once, then play it back many times including the dialog interaction with the server. In a generic example, this could allow one to record "metabookmarks", i.e. bookmarks that record not only a destination Uniform Resource Locator ("URL"), but also the required steps to navigate thereto, and play them back.

Second, an extraction API 20 enables an application to robustly define data segments in a web page. As for the navigation module 10, there are recording and playback modules 22 and 24. However, instead of recording/playing across pages and web space, the extraction module 20 records and plays across elements within a single page. Artificial intelligence ("AI") techniques can be utilized to enable pattern matching to ensure that the relevant information will still be retrieved even if the page is modified (within reasonable limits, of course). For instance, if a given numerical value such as a stock value appears at a certain location in a document, the present invention enables an

application program to retrieve it many time by playing the extraction instructions, even if its location changes because banners have been added on top of the page for example. Other changes like font, color, and size can be handled as well. Moreover, the present invention is preferably capable of performing some degree of learning when presented with dynamically generated pages (program generated pages), like pages containing stock quotes, weather data, for example. From a few examples (preferably two), the extraction module 20 infers extraction rules and applies them to the rest of the data in the page. This is especially significant since the logic of the data organization is usually hidden to the user.

According to the present invention, data extraction rules are preferably kept separate from the extraction program itself, making it possible to update them separately. Utilizing the present invention, data from different sites can be gathered for simultaneous display, in formats like MS Word, MS PowerPoint, or MS Excel, for example, or for further processing according to the each user's particular needs, i.e. extraction of statistics, computations, etc. In other words, the navigation API 10 provides two services: recording of navigation paths 12 and playback of that which has been recorded 14 while the extraction API 20 provides two services: recording of extraction patterns 22 and playback of extraction patterns 24.

Referring now also to Figure 2, and first to the navigation recording module 12 which records navigation paths and steps, each HTML document or page is provided 30 to the navigational module 10 by either a web application calling the present invention (for example as a mini-agent), or a web browser such as Microsoft Internet Explorer (IE) for example. Each provided HTML document enters the navigation module 10, and is

converted 40 into an object hierarchy. As is well known, referring for example to Internet Explorer for sake of explanation, when an HTML document or web page is requested, the page is transmitted from the server to an intermediate receptor, as a data stream, a well known example being Winsock. The data stream is separated and arranged into a plurality of elements configured in a well-defined hierarchy so they may be used to build the actual document displayed to the user by Internet Explorer. Hierarchy objects or elements include lists, tables, images, and columns for example. Further, these elements can be nested to provide for multiple levels, for example tables within tables and so on. As has been set forth, the HTML document itself can come from either browser controls embedded within an application, or from a web browser itself. This object hierarchy is cached and used during the recording 50.

Once the page has loaded, a user interacts 60 with the loaded page by clicking on or activating links or buttons, entering data and so on as is well known. The navigation recording module 12 traps 70 each user generated event 60 (such as clicks and keyboard inputs) at the HTML level using the object hierarchy built previously (40 and 50). Thus, events are not trapped at the screen level, making the recording immune to particulars of the current desktop organization, but rather defined relatively within the page based upon the object hierarchy (i.e. using its lineage).

During the recording, navigation module 10 memorizes which anchors were clicked and which forms were submitted 70, and maps 80 each user event to an element in the recorded HTML object or element hierarchy. This information is preferably stored 90 in an Extensible Markup Language ("XML") file for example, although any suitable file format could of course be utilized. It should be recognized that the use of an XML format

insures portability, readability, access to the XML object model, and the capability to programmatically modify the recorded path transfer though. The recorded XML file preferably contains tags indicating the navigation steps and parameters entered on forms. The navigation XML file contains details about the recorded navigation in a format that can be read by the player 14. More particularly, the navigation XML file preferably includes the series of steps that correspond to different web pages that are loaded during the recorded initial navigation process. Each step includes information about the page and a collection of elements that correspond to HTML elements like hyperlinks and form fields that are acted upon. Each element includes information on how to locate the corresponding HTML element and what action to perform on or with it. Preferably each navigation step is recorded as an XML entry in the input file. Generally, there are two different types of steps for playback by the playback module 14: form variations and URL variations.

Referring first to form variations, XML encoding of a form provides key-value pairs for form parameters. These can be changed by an application, either at the XML file level by simply replacing text, or at the playback level by accessing the playback module 14 and specifying new form parameters to replace the ones originally recorded using the recording module 12. This enables an application to automatically repeatedly query a site while introducing variations. For instance, a form can be filled to get pricing information from an online bookstore for different titles by changing a single parameter in a query form. According to the present invention, form variations can be automatically used to accomplish this task. This, of course, represents a significant improvement over the prior

art as multiple queries can now be automatically run based upon a single exemplematic query.

Referring to URL variations, these can similarly be applied to recorded URLs but with even more possibilities. The simplest example is the straight encoding of a URL. As is the case with form parameters, URLs can be replaced by others either at the XML file level or by direct access to the playback module 24. The playback module 24 can also understand some more flexible ways of specifying anchors. An anchor can be specified relatively to the structure of a document as will be discussed. For example, the player 24 can for instance use specifications like "the third anchor in the document". Also, the playback module 24 can also accept specifications that are text-based, such as the first anchor which contains the text "IBM".

The navigational playback module 14 uses the XML encoded navigation paths (those created during the recording phase 12 with the option of introducing multiple variations) as input files. It can then reproduce the navigation path by automatically generating clicks 100 on documents and/or submits 110 on forms specified relatively to the structure of the document (referring again to the object or element hierarchy map). Again, the API's 10, 20 can use specifications like the "the third anchor in the document" and specifications in the content space (text-based) such as "the first anchor that contains the word "IBM" to identify particular page elements.

Referring now also to Figure 3, as will be evident to one possessing ordinary skill in the art, there are generally three forms of links: hardcoded or static links, structure-based links (such as the 35th link from the top), and text-based links (such as that demonstrated for "IBM"). According to the present invention, form elements are recorded

as opposed to the "form" itself to provide robustness. The player 14 preferably looks for these elements in a page, then goes back to the parent form and submits them. In this way, forms can be moved around in a page that contains them, and yet the playback module 20 will still be able to identify them. Out of security considerations, the navigation files are preferably encrypted since they may contain login and password information. A calling application can provide the necessary interface for variance of extraction rules. For instance, if an application allows for automatic navigation to a set of password protected sites and is intended to be used by different users, a calling application should query these users for individual passwords to provide to the playback module 20. In summary, a navigation includes going to a starting page, performing a series of actions such as generating clicks on hyperlinks and submitting forms which in turn cause a target or final page to be loaded. The final or target page being loaded represents the final step in a navigation sequence that is stored as a navigation file.

Referring now also to Figure 4, it should be understood that while the navigation module 10 operates across multiple pages or web space, the extraction module 20 operates at the HTML or web page level. Similarly to the navigation module 10, the extraction module 20 preferably includes a recording module 22 and a playback module 24. The extraction module 20 preferably follows the navigation module 10 by processing each final or target page loaded in response to the navigation module 10 to extract relevant information, or data therefrom.

Referring first to the extraction-recording module 22, it accepts as input text selections within an HTML page. A pattern must be identified from these selections (preferably two, however as more are considered the pattern will become more predictable

as is well known) according to which data will be extracted using the player 24. According to the present invention, three different techniques can be used for identifying a pattern according to which data can be automatically extracted from a page. The first is structure or DOM driven, the second is content or text driven and the third is a combination of these criteria. Structure driven pattern generation relies upon the underlying structure of a page to identify elements, i.e., uses their inter-relation and relation to the page as ā whole. Content driven pattern generation relies upon certain key phrases that are present in each element of interest or in another element having a known relationship to an element of interest. While, the combination approach uses both of these approaches to identify the pattern. Referring now also to Figure 5, a user selection (for example all HTML elements included between a start position of the cursor and an end position, i.e. a portion highlighted) is expressed in two different spaces: first, the structural space based on the document's hierarchy, and second, the contents space based upon the documents text (individual portions of an object defined within the HTML object hierarchy). Thus, a selection in a page will be expressed in structural space as a certain cell in a certain row of a certain table for example, while in content space as the n^{th} word in the m^{th} line of the p^{th} section. Quick access to relevant information usually takes both specifications: For instance, an array cell could be a leaf in the structural space, the smallest accessible element of which can contain a block of text. To represent the selection of one word within this block, the contents space specification is needed. The module uses the definitions in these spaces of a selection and processes the selection in an array, list, or other similar configuration.

Additionally, the recording module 22 is capable of inference: i.e. given two selections, a pattern is extracted in a structural space by going up in the document hierarchy and retrieving the siblings using conventional AI techniques. A pattern can be extracted in the contents space by generating a regular expression which matches the selections. The pattern extraction is preferably made available to the application through the API 20. Referring again to Figure 4, after a page is loaded 140 the web page is parsed 150 to produce 160 the HTML hierarchy similarly as has been set forth for the navigation script, the user would then make a selection from the page by highlighting that portion 170. Regarding the example presented in Figure 4, a user selected the starting element portion AA, part of the parent element A, and ended his selection on element portion BB of element B. That selection is then mapped 180 into the HTML hierarchy that was created 160. In other words, the user started in tag 1 of content "AAA" and ended in tag 2 content "BBB". The system next maps 190 selected lines and words to lines and word offsets within the selected elements, thus generating the start cursor position within tag 1 and the end cursor position within tag 2. In other words, the users selection is mapped 190 to generate the recognition that the user selected tag 1 from line 1 starting at word 2, to tag 2, line 1 ending at word 2, for example. Finally, the extraction script is again recorded 200.

The output of the recording module 22 is again preferably an XML file describing the selection(s) to be given to the playback module 24. It should be noted that this selection specification is expressed in an abstraction of the actual page it was recorded from, i.e., it can be applied to another page with similar structure. As will be evident to one possessing ordinary skill, this is a powerful feature that allows an application program

to extract information from pages similar to the recording page as well as from future iterations of the recording page.

Referring now to figure 6, the playback module 24 uses the retrieval specifications from the recording module 22 (using the XML file) to extract actual information from a final page loaded in response to the navigation playback module 14. The retrieval information contains both the structural and contents space specifications, and if these are expressed as patterns, a termination condition (for example, the first 5 matches, the last one, etc.). The output of the module 24 is a text range 210, i.e., a subset of the HTML code of the target page that matches the retrieval, i.e. extraction specifications. In order for the extracted text range to be printable, the module 24 extracts its lineage (parent structure, grandparent, etc.) which specifies its context. The extracted text range can represent pure text information, but also pictures, charts, etc. A calling application can arrange these elements in a form suitable for display to the user, or for future processing. The extracted text range embeds information from both the structural and contents spaces, and preferably maintains links which permit further processing in both. As set forth, according to the present invention techniques of pattern recognition and extraction used include: a structure or DOM based method, a content based method and a criteria or a combination of structural and content based method.

Referring now to Figure 7, the core technology according to the present invention is made available through the navigation 10 and extraction 20 APIs. Thus, a wide variety of applications can be built on top of the core technology. Development of a general application 220 taking advantage of the core technology would, for example, identify a set of pages relevant to the chosen domain, build the navigation paths to these sites using the

recording module 12, build the interface to get the user's personal information (login name, password, etc.), if needed; build the information retrieval rules to be used by the extraction API 20; build the interface to present the retrieved information to the user; and build the interface to refine the information retrieved using the inference module.

An example utilizing this technology is illustrated by the following. Referring now also to Figure 8, a user needs only to start recording using the module 12 and "surf" as usual to the desired destination URL using Internet Explorer for example. Upon reaching the destination URL the user indicates the same to the recording module 12. The user can interact with the module 12 using buttons 230 for example. Referring now also to Figure 9, the user then highlights (i.e. using a mouse drags over) the desired information to be extracted 240 and the highlighted data is automatically recorded and displayed in lower window 250. The user may now have the option of editing the extraction script to fine tune his selection. Preferbly, a second selection is highlighted by the user so a pattern can be generated using conventional AI techniques. Referring now also to Figure 10, the user, upon playing the recorded Navigation and Extraction scripts (collectively referred to as Navex) can automatically extract numerous records 260 which match a pattern (which may or may not have been fine-tuned) derived from the extracted text by the recording module 22. Referring now also to Figure 11, a user can utilize the "find text" command 270 to find a particular string, such as a stock symbol and extract data according thereto. In the case of Figure 11, the account type, symbol, description, quantity, high, low, change, last price and market value. This represents the use of content driven patterns, the steps of which are recorded by the extraction recording module 22.

Structure based recognition and extraction relies on the fact that a web page is a collection of HTML tag elements that are typically arranged in a repetitive manner. If the HTML pattern is converted to a "road map" where certain elements are defined with respect to the top of the page, a clear structure emerges. According to the preferred embodiment, the user is required to define two instances of his perceived pattern. The two selections are then compared to determine what structural commonality exists. For example, a table can include cells, each cell having a well-defined structured relationship to the parent container, the table. Alternatively, the user could use the object or elementm ap by highlighting portions thereof, the selection of which is preferably mapped to the appropriate displayed element. A user could then define the desired pattern dependent upon the elected map element by defining which portions are to be varied, and how to vary them. Contents based recognition and extraction relies upon locating key words in a web page then selecting tags that contain that key word. A regular expression search is preferably used to define complex text pattern searches. For example, in the table, all rows containing "300L" or '200L" can be extracted using a regular expression: "[23]00L". In other words, the selection is based on the contents of the tags, not their relationships to the table. Criteria based recognition and extraction relies upon the fact structure and content based matches operate in two distinct domains. Criteria matches bridge the gap. In addition to both structure and content consideration, criteria matches can also include presentation attributes such as color, X and Y location, font size etc. The pattern selection process is a logical AND of all specified criteria. In other words, criteria extraction techniques can be used to recover all cells in Column 3 (section based) that require that the cell in column 7 have a red font (attributes) and contains a minus sign (contents based).

Referring again to structure based recognition and extraction and also to Figure 12, therein is illustrated another example. Therein, a user has selected text 1110 within a conventional web page 1120. The parental hierarchy of this highlighted element within the page 1120 can be seen as DL-DT-A (1130-1140-1150). According to the present invention, the system infers that the user wishes to further extract information 1160 using this parental hierarchy information (DL-DT-A, 1130-1140-1150). Referring now also to Figure 13, therein is again illustrated highlighted element 1110 and related parental hierarchy information 1130-1150. Further illustrated are element 1160 and related parental information 1170 - 1190. Using AI techniques, the present method evaluates parental information 1130 - 1150 and 1170-1190 to determine a most likely pattern which the highlighted elements 1150 and element 1160 can be classified as satisfying, and then finds the next element which matches that pattern, if any.

Again it should be understood, a web page is a collection of HTML tags. Some common tags are: TABLE - a table of data internally consisting of: TR - a Table Row which is turn contains, TD - a "cell' of data, A - Anchors and P - Paragraphs for example. Formatting tags are also typically included such as: B - Bold, I - Italic, U - Underline and BR - Break (new line). When two selections are made the HTML pattern extractor determines what the two selections have in common. At the least: the two patterns are the same TAG and the "Lineage or ancestry tree match: i.e., both selections have ancestry that matches: DL-DT-FONT A. If the pattern type and lineage match, then a pattern is determined to exist. It should be understood a lineage may be "clouded" by formatting elements such as B, I, U. The pattern extractor preferably removes these from the considerations, i.e. ignores them, when performing the match.

Taking a moment to review parental information 1170-1190 related to element 1160, and parental information 1130-1150, pattern DL-DT-A is clearly recognizable and identifiable using convention AU techniques. (Please note in this case 1170 and 1130 refer to the same DL). Further, series element 1200 associated with the object hierarchy of element 1140, increments according to the record number when compared to element 1210 related to parental hierarchy element 1180 (1 to 3). Accordingly, software implementing the present invention preferably uses conventional pattern matching techniques well known to those possessing ordinary skill in the art to infer the next record the user wishes to extract should have a parental hierarchy which fits the pattern DL-DT-A and includes a value of 5, then 7, 9 ... associated with the DT parental hierarchy element. Having been taught the pattern (DL-DT-A1..3..) through the user's interaction or from an application calling the present method, the navigation and extraction APIs 20, 40 can be used to extract numerous records matching the pattern defined.

The purpose of "find text" and hence content based pattern recognition is to provide answers by utilizing text based searches. For example the regular expression: [0-9]*[0-9]*[0-9] + \ / [16\|32]* will return any number of any length followed by any number of any length followed by any number of any length and/or a number divided by 16 or 32. As will be understood by those possessing ordinary skill in the art, this expression is particularly useful in extracting stock quotes from a page. The final text command is simply recorded as a step in the extraction file. Information extracted can further be mapped according to known relationships, after which application specific components can be built to permit the use of standard query tools for example.

As set forth, content based pattern matching uses a selection process by conducting a regular expression search on a page for a pattern: i.e., [23]00L. Elements that contain the pattern are tagged. Tagging is a process of marking a HTML element in the HTML DOM (structure map) as passing the key word search filter and being of interest to the user. Next, for each tagged element, the user may select an element before or after the selection by traversing the tree. He may do this in two ways: moving up to the parent element (Up Parents), shifting the source Index +/- (Shift offset), or defining the road map that specifies a set of directions to go from the tagged element to another related element. For example, a compound road map is <UP>TR:TD,TD,TD. In English: move up to the first TR then go to the 3rd cell in that row.

Criteria based matches identify selections based on a series of tests. DOM or structure based matches require well structured web pages or the ability to analyze the page and set stop and ignore tags accordingly, i.e. start at the third row in the second table and ignore formatting tags while content based matches are less stringent. If the key word search succeeds, data is returned if the key word exists in all items of interest and the user is interested only in the key word and not its location in the DOM. Criteria based matches are a combination of both types of matches. In a typical situation, the criteria for selecting an item will be based on: (1) its structure attributes (tag name, lineage), (2) its presentation – font type, location on the page, height, width etc. and (3) content in both the HTML text and the text shown to the user ("Innertext"). All tags in the Web page are preferably examined to determine whether they meet the selected criteria.

For example, and referring now also to Figure 14, regular expressions may be entered for lineage or the text contents (items 2,3,4,5). Additionally ranges for the

geometry may be entered in items 7,8,9,10: the Left, Height, Top and width of the tag element. The tag elements that qualify are those that meet all the criteria specified. More particularly, according to preferred form of the invention, the following selection criteria are utilized: (1) Tag name: - Select all tags based on a list of tag names, typically one item, (2) Lineage - being computed by going up the DOM tree till you reach the BODY (top of the document), the expression entered has to match a portion of the computed lineage or entirely. (3),(4) Text - search for text in either HTML or the inner text, (5),(6) tag inclusions - lineage may not be specified, instead an include/exclude list may be. (7),(8),(9),(10) Geometry - Top, Left, Width, Height. Items 1,2,3,4,5,6 are computed by the system based on the DOM tree and Innertext and InnerHTML attributes of the element (available from the DOM tree). Items 7,8,9,10 are attributes of the element, available in the DOM tree.

From a general standpoint, these individual pieces can be used in a total system solution as well. For example, navigation and extraction scripts (collectively Navex scripts) can be read and executed. If one fails, other scripts can be called to eventually result in a fail condition or the extraction of relevant data from a web page. Two tables can be generated from the extracted information, the HTML descriptor and the actual text. It should be understood that the HTML descriptor for the extracted text is important because it may be necessary to fully understand what has been extracted, for example green for stock prices which have risen, those which are positive, and red for those which have fallen or are negative. An application can then cross reference and use this information to permit a user of it to have access to the information in a database format.

The power of the pattern matching system according to the present invention lies in teaching patterns of any of the three types and the system automatically generating the same required Snippet object structure for all cases. Referring now also to Figure 15, all three pattern-matching systems "meet" at Macro steps. That is, the rest of the system receives a pattern of type Snippets, on which subsequent operations are performed. Having recorded navigation and extraction schema in XML files for example, scripts can be written to employ them. A representative script could for example, automatically navigate to a web page, extract the information on the final web page, process the data, including validations and return results as an HTML table in the browser by executing a series of operation according to a navigation file, then an extraction file, then performing operations on the extracted data. The constituents of the script include one or more navigation and extraction schemas and one main program that calls these schemas in the order required. There may also be other programs called by the main program to perform specific processing. A script helper is preferably provided to: define the schema files you want to reference in your program (akin to an include statement in C), write the glue to call these programs in a main program module, include references to objects such as databases, files etc that you want to talk to, present the output data to an IE browser or send an email notification.

The input to the script helper is preferably an XML file that defines the constituents of the complete program. Each XML step is a file to be included in the final program. Based on file extensions, the system preferably will automatically convert the File input to VBScript code. Each conversion results in a Subroutine (or function) being added to the

main program. The main program can now call the subroutines to perform automated navigations and extractions.

Default execution command lines are stored in the XML make file while file extension type informs the system what type of file is being loaded and defines subsequent processing. The system according to the preferred invention preferably has objects loaded which include an extraction run time processor which knows how to run a taught, schema, gives access to internal objects such as the CEFIND, CECRITERIA and documentation on how to use their functions in the Object Browser of VB; a grid processor which processes the extraction data into formats requested and provide low level presentation capability; and object schema which takes the snippet grid information and provides XML/Excel/database access. These objects are preferably accessible from within scripts loaded to the player.

Referring now to Figure 16, Web Bands is a software system that is preferably installed on a user's computer and exposed as standard (Vertical and Horizontal) bands housed within the browser, for example Internet Explorer. The Vertical Band, which is the main user-interface, contains items (like buttons or hyperlinks) that allow the user to perform specific tasks using the underlying technology according to the present invention (for automated Navigation, Extraction, etc). The Horizontal Band provides an additional interface area for specific applications (that would be invoked from the Vertical Band). The Web Bands house HTML pages that act as the user interface for the system.

To install the system, a user would go to a website and download an Installer file to his microprocessor based device, i.e. personal computer. The user would then execute this downloaded installation file by double-clicking the icon or executing the Run

'Download Folder\install.exe' command at the command prompt for example. This would start the process of installing the Web Band system on the user's computer. The installable is preferably an Active Setup, which means that the Installer downloads the various files that are installed on the user's machine from the web site. This ensures that the latest version of the software is always installed on the user's computer. It should be understood, the user's computer must be connected to the Internet or another network through which it can access required data from the website. Also, the entire installation may be distributed on removable media, like CD-ROMs. The installation creates the necessary folders required by the Web Band system, installs required files in these folders, registers the various COM components on the target computer and modifies the system registry to register the Web Bands as IE bands.

Once the software is installed, the user can view the Web bands by starting IE and selecting a View - Explorer Bar - VerticalBand menu option for example. The entire installable is created using commercially available software such as Wise Installer for example. The system includes the following major components: Web Bands, the component that an end-user interacts with in IE, and implements the specified COM interfaces required to be Explorer bands. Associated file(s): NNEBand.DLL, IEBand.OCX, IEBand.INI. A Web Player a component embedded in the Web Band to provide the main underlying functionality to download and execute scripts that constitute a task. The Web Player itself contains many functional objects like the Web Navigator and Web Extractor that can be used by scripts. And, Band Aid, a component embedded in a Web page (using the OBJECT tag in HTML) to connect to the Web Player, so that VBScript and Javascript within the HTML page can make use of the Web Player

functionality. For eg. A Band Page uses the Band Aid object to get a handle to the Web Player in the current IE instance. VBScript or Javascript within the Band Page then uses the Web Player to execute a script that performs a certain task.

The Web Bands have corresponding Web Pages (HTML files) that constitute the user-interface. These files are specified in the INI file for the Web Bands. In the default installation, they are specified as web pages on the web-site. A corporation or users can author their own web pages and customize the look and feel of the Web Bands to present a rich user interface. These pages can be created in a standard HTML Editor and can contain DHTML, VBScript/Javascript, applets, plug-ins, etc.

Some guidelines need to be followed to use the underlying Web Player system. The user can modify the INI file entries to specify the required Band pages as default. The Band Pages consist of HTML Elements (like Hyperlinks or Buttons) that the user selects/clicks on to carry out specific tasks. All the tasks correspond to scripts that are executed in the Web Band at run-time. Thus, each Element (that constitutes a task) in the Web band has some VBScript or JavaScript code that is executed in order to run the corresponding application script. The Web Player 24 (which is a part of the Web Bands) provides the functionality to download scripts and execute them on the user's computer. This functionality can be invoked by standard VBScript or JavaScript in the Band Page.

According to a preferred embodiment of the present invention, the provider also wants to promote a developer community that would write application scripts for the Web band system. The scripts would be made available on a web site, with some kind of collaboration with the developers. Once a developer registers with the provider, script generation tools like the Web Recorder system would be available to him for downloading.

The provider would provide developers with working space on the web site to upload the scripts. These scripts might be checked by the provider for potential errors or security hazards and then made available to an end user (with the Web Band system). To write a script for the Web Band system, one does not need to be an expert programmer. The developer should have a fair knowledge about VBScript, HTML and read about the structure and functionality of the Web Player. An end user would be able to customize his Bands in order to include new scripts made available on the web site. The provider would preferably provide a simple mechanism on the web site, to allow a user to customize the appearance and functionality of his Web Bands. After the user saves the customized band, the customized Band will be displayed whenever the user views the Web Bands.

In a preferred embodiment of the present invention, scripts are stored at a centralized repository that is accessible through the Internet for example. In this way, if there are multiple users of a script and should that script fail, it is easy to ensure each of the users have a corrected script as soon as possible, i.e. as soon as it is downloaded to the central repository. One can activate a script by requesting access to it, temporarily storing it and then locally running that script.

It should further be understood, no matter how robust the extraction and navigational methods utilized according to the present invention are, they will sometimes fail, either because changes of too great a magnitude have been made to the destination or intervening web pages or web pages are no longer accessible for example. Accordingly, it is desirable to have some way to audit or confirm that navigation and extraction scripts are still operational. If the scripts are stored in a central repository as discussed supra, auditing their correct operation becomes considerably easier. Generally, by periodically accessing

each script and at least partially executing it, one may determine whether it is functioning properly by comparing the extracted data against an expected result. For example, if a stock price is intended to be extracted, it is expected that the data extracted from the destination page should be a number, and probably a number ending in some fraction. If the data extracted does not fulfill this expectation, i.e. instead includes alphabetic data, then it is known the script has failed. The script can then be automatically disabled, and proper notifications sent to individuals or entities responsible for the operation of the failing script by e-mail or pager notification, for example. Alternatively, a failing script could be re-accessed one or more times, or at predetermined intervals, to determine whether it is now operating correctly and whether the error that apparently caused the script to fail has seemingly ceased to cause a problem. In such instances, the script could be conditionally reactivated depending upon design criteria. Such design criteria may include conditional limitations such as whether a technician has had an opportunity to review it, and whether it has failed before. In the preferred embodiment, such an auditing of scripts can occur many times a minute for some portions of the scripts (such as accessing a price of an item for sale from a plurality of vendors websites) while only executing other portions of scripts considerably less frequently (actually buying some of those items for purposes of auditing the remaining portions of the script).

Although the present invention has been described with a particular degree of specificity with reference to a preferred form of the embodiment, it should be understood that numerous changes both in the form and steps disclosed could be taken without departing from the spirit of the invention. The scope of protection sought is to be limited only by the scope of the claims which are intended to suitably cover the invention.